An Evolutionary Pentagon Support Vector Finder Method

Seyed Muhammad Hossein Mousavi[a]

[a]Department of Computer Engineering, Buali Sina University

District 2, Hamadān, Iran

Email: mosavi.a.i.buali@gmail.com

Vincent Charles[b,*]

[b]School of Management, University of Bradford

Bradford BD7 1DP, United Kingdom

Email: c.vincent3@bradford.ac.uk

Tatiana Gherman[c]

[c]Faculty of Business and Law, University of Northampton

Northamptonshire NN1 5PH, United Kingdom

Email: tatiana.gherman@northampton.ac.uk

* Corresponding Author

An Evolutionary Pentagon Support Vector Finder Method

Abstract

In dealing with big data, we need effective algorithms; effectiveness that depends, among others, on the ability to remove outliers from the data set, especially when dealing with classification problems. To this aim, support vector finder algorithms have been created to save just the most important data in the data pool. Nevertheless, existing classification algorithms, such as Fuzzy C-Means (FCM), suffer from the drawback of setting the initial cluster centers imprecisely. In this paper, we avoid existing shortcomings and aim to find and remove unnecessary data in order to speed up the final classification task without losing vital samples and without harming final accuracy; in this sense, we present a unique approach for finding support vectors, named evolutionary pentagon support vector (PSV) finder method. The originality of the current research lies in using geometrical computations and evolutionary algorithms to make a more effective system, which has the advantage of higher accuracy in some data sets. The proposed method is subsequently tested with seven benchmark data sets and the results are compared to those obtained from performing classification on the original data (classification before and after PSV) under the same conditions. The testing returned promising results.

Keywords*: Big data; data mining; support vector; Artificial Bee Colony (ABC); evolutionary clustering; Fuzzy C means (FCM); Pentagon Support Vector finder (PSV).*

1. Introduction

The big data era has brought many challenges with it (*e.g.*, Charles & Emrouznejad, 2018), which deemed the traditional data processing applications too weak to deal with them. Among these challenges, we can mention networking, capturing data, data storage and analysis, search, sharing, transfer, visualization, querying, updating, and information privacy (Charles & Gherman, 2018; Charles, Tavana, & Gherman, 2015). In light of the above, big data is essentially about the complexities of the data that we attempt to unpack (Charles & Gherman, 2013) and in this sense, accuracy and speed in computation are two very important aspects to consider when processing these large amounts of data. But when we are dealing with classification tasks, most of the data may turn out to be outliers and, thus, represent unnecessary data points. Detecting outliers, or in other words, those data points that do not conform to expected behaviour, has attracted a lot of attention due to its applicability in a wide variety of domains (Rekha, Abdulla, & Asharaf, 2017).

In order to deal with such problems, techniques called *support vector finders* have been developed to identify and remove outlier data. These techniques help to decrease the classification computation speed significantly. Also, with inspiration from natural phenomena, it is possible to make much more robust algorithms and methods, improving their performance and helping to understand the dynamics in the data better. Support vector finders are very important in data mining and big data. It is beyond the purposes of the present paper to explore such territory, but just for the purposes of highlighting the practical relevance, here are some of the fields which benefit from these methodological advances: data mining has application in a variety of settings, such as industry (Piatetsky-Shapiro, 1999), healthcare (Koh & Tan, 2011), security (Lee & Stolfo, 1998), and medicine (Bellazzi & Zupan, 2008), among others; and big data

has application in fields such as biomedical research and health care (Luo, Wu, Gopukumar, & Zhao, 2016), international development (Global Pulse, 2012), manufacturing (Lee *et al*., 2014), internet of things (Morris, 2014), and so on. For a comprehensive discussion regarding more societal benefits and current uses of big data, the interested reader is referred to the recent work by Emrouznejad & Charles (2018).

When dealing with data mining, researchers have been met with the outlier issue in classification problems, especially when the data points are very close to each other. Let us imagine we would like to classify the cat species based on tail size. The sizes of the cats' tails are different, but in many cases, they are also similar; hence, misclassification can occur. This problem emerges when the borders between classes are very narrow or when some samples of classes are mixed. Indeed, some classification algorithms could extract these mixed samples; nevertheless, the far samples or outliers will affect the calculations for that specific algorithm. To fix this problem, researchers have had to remove some data or use clustering techniques to get rid of outliers, both of which, however, have shortcomings. On the one hand, removing outliers may cause the removal of some samples from other classes, which are not outliers for that particular class. On the other hand, clustering techniques change the real position of the samples in the space. Clustering algorithms are optimized to find clusters rather than outliers and a set of many abnormal data objects that are similar to each other would rather be recognized as a cluster than as outliers; furthermore, the accuracy of outlier detection depends on how well the clustering algorithm captures the structure of the clusters. Our proposed approach avoids such shortcomings and leads to the preservation of the real data without outliers.

Detecting the position of these outliers in the data space and removing them for better processing demands the use of intelligent algorithms. Also, as these data (in view of their position in the data space) are statistical data, the output of a support vector finder system could be used as an expert system (without human interaction) for faster and automated processing and free from any error in the classification task.

In the present paper, we aim to contribute to this research strand (i.e., eliminating outliers) by introducing a unique approach for finding support vectors, which consists of four steps: (1) reducing the amount of data using evolutionary clustering (Artificial Bee Colony together with the Fuzzy C Means-Manhattan method), (2) labelling the remaining data using the K-nearest neighbourhood classifier, (3) removing outliers based on a specific threshold, and (4) calculating the area of the pentagon and the angle between samples of existing classes to determine the position of the final support vectors. The originality of the current research lies in the use of geometrical computations and evolutionary algorithms to make a more effective system. Furthermore, our approach has the advantage of high accuracy in the data sets.

The remainder of the paper unfolds as follows: Section II provides a description of some basic definitions and concepts and section III explores existing research on the topic. Subsequently, Section IV introduces the proposed support vector finder method and related process in detail. Then, the proposed method is applied to few benchmark data sets for validation purposes and the results are presented in Section V. Section VI concludes the paper with relevant discussions and suggestions for future lines of research on the topic.

2. Main definitions

2.1 Evolutionary Computation (EC)

Evolutionary Computation (EC) is a well-established branch of computer science, in which highly efficient optimization techniques are inspired by nature and biological evolution (Spirov & Holloway, 2013). As such, this kind of computations are inspired by evolutionary principles for automated and parallel problem-solving (Bäck, Fogel, & Michaelewicz, 1997; Eiben & Smith, 2003; De Jong, 2006) to solve a broad range of complicated mathematical problems that are challenging for traditional computational methods. There is evidence that many practical problems can be solved efficiently using EC (Zhu, Bastern, Geilen, & Stuijk, 2012).

To do this, EC processes start with a pre-defined number of populations in different number of generations to do mutation and recombination during a pre-defined number of iterations. The goal is to achieve the highest fitness function or the lowest cost function, depending on the need. During generations, individuals with a better cost or fitness function are saved and the less desirable solutions are eliminated. This process continues to evolve and acquire more optimized individuals, which are the best final solutions of the mathematical problems (Bäck, Fogel, & Michalewicz, 1997). EC is a much advantageous approach, as it requires only a suitable coding structure which can modify the solutions during reproduction and a system for assessing the quality of individual solutions (Yagain & Vijayakrishna, 2015). At the same time, however, it does not need any prior knowledge of solution search space (Parhi, 2007).

An algorithm which uses EC to solve a problem is called an Evolutionary Algorithm (EA). Some examples of EAs are the Genetic Algorithm (GA) (Mitchell, 1998) (which is considered to be the most biologically accurate EC model, as well as the most popular EC instance due to its intuitive usage and ease of implementation (Drugan, 2018)), ant colony optimization (ACO) (Dorigo, 1992), particle swarm optimization (PSO) (Kennedy, 2017; Kennedy & Eberhart, 1995; Mavrovouniotis, Li, & Yang, 2017), differential evolution (DE) (Storn & Price, 1997; Das, Mullick, & Suganthan, 2016), Cuckoo search (Yang & Deb, 2009), and Big Bang-Big Crunch algorithm (BBBC) (Erol & Eksin, 2006), among others. In general, these algorithms have gained increasing popularity overall thanks to their ease of implementation and exemption from the obstacle of derivative (Liang, Hu, Zhu, & Zhu, 2017). Furthermore, these algorithms have applicability in a wide range of domains, such as city planning (Balling & Wilson, 2001), robotics (Yu, Jinhai, Guochang, Rubo, & Haiyan, 2002), industry (Takagi, 2001), games (Gillespie, Gonzalez, & Schrum, 2017; Justesen & Risi, 2017), control (Parker & Nitschke, 2017; Reed, Hadka, Herman, Kasprzyk, & Kollat, 2013), and more (Fogel, 2000). For further information, the interested reader is referred to the studies by De Jong, Fogel, and Schwefel (1997) and Bäck, Fogel, & Michaelewicz (1997) for a history on the beginnings of EC and to Kallel, Naudts, and Rogers (2001) for a discussion on theoretical aspects of EC.

2.2 Artificial Bee Colony (ABC)

Artificial Bee Colony (ABC) evolutionary algorithm is a relatively new optimization method proposed by Dervis Karaboga in 2005 (Karaboga, 2005) and inspired from the intelligent foraging behavior and information sharing capability of the honey bee swarm (Karaboga, Gorkemli, Ozturk, & Karaboga, 2014; Bansal, Sharma, & Jadon, 2013). Developed initially for continuous optimization problems, it was later modified to solve discrete optimization problems, as well (Kashan, Nahavandi, & Kashan, 2012; Kiran, 2015) and has since given birth to various ABC algorithm variants (Alatas, 2010; Akbari, Hedayatzadeh, Ziarati, & Hassanizadeh, 2012; Karaboga & Akay, 2011; Duan, Xu, & Xing, 2010). This algorithm is one

of the most novel and robust algorithms, among other famous evolutionary algorithms, and converges with good speed in global maxima (based on input parameter and problem type). It has proven its applicability in areas such as neural networks (Kumbhar & Khrishnan, 2011), forecasting stock markets (Hsieh, Hsiao, & Yeh, 2011), image processing (Draa & Bouaziz, 2014), assignment problems (Metlicka & Davendra, 2015), structural engineering (Ding, Huang, & Lu, 2016), and more recently network topology design (Saad, Khan, & Mahmood, 2018), among others.

According to Karaboga (2005), the ABC algorithm involves three types of bees that cooperate with each other in performing different tasks (known as *division of labour*): employed bees, onlookers, and scouts. Employed bees are associated with a particular food source that they are currently exploiting and exchange information about this particular source with its neighbours. The number of employed bees in the colony is equal to the number of food sources around the hive. Furthermore, the number of employed bees and onlooker bees are equal. Once a food source is abandoned by its employed bee, the employed bee turns into a scout and searches for new food sources. Onlookers wait in the nest and watch the dances of the employed bees (called *waggle dances)*, which they use to then choose the food sources on the basis of the information shared by employed bees with regards to the quality of the food sources.

The ABC algorithm generates a randomly distributed initial population of SN solutions (food sources), wherein SN denotes the swarm size. $X_i = \{x_{i,1}, x_{i,2}, \dots, x_{i,n}\}$ represents the i^{th} solution in the swarm, where n is the dimension size. Each employed bee X_i generates a new candidate solution V_i in the neighbourhood of its present position, as indicated in the equation (1) below:

$$V_{i_k} = X_{i_k} + \Phi_{i_k} \times (X_{i_k} - X_{j_k}) \quad (1)$$

When X_j is a randomly selected candidate solution ($i \neq j$), k is a random dimension index selected from the set {1, 2, …, n} and Φ_{i_k} is a random number within the interval [-1, 1]. Once the new candidate solution V_i is generated, a greedy selection is used. If the fitness value of V_i is better than that of its parent X_i, then we proceed to update X_i with V_i. Otherwise, we keep X_i unchanged. After all employed bees complete the search process, they share the information of their food sources with the onlooker bees through *waggle* dances. An onlooker bee evaluates the nectar amount information taken from all employed bees and chooses a food source with a probability related to its nectar amount. The probabilistic selection is a roulette wheel selection mechanism (Fogel, 1997). If a position cannot be improved over a predefined number (called limit) of cycles, then the food source is abandoned. Let us assume that the abandoned source is X_i and then the scout bee discovers a new food source to be replaced with j^{th} as in the equation (2) below:

$$X_{i_k} = lb_j + rand(0,1) \times (ub_j - lb_j), \quad (2)$$

where rand (0, 1) is a random number within the interval [0, 1] based on normal distribution, and lb and ub are the lower and upper bounds of the i^{th} dimension, respectively. Figure 1 shows the process of the ABC algorithm in a flowchart form.

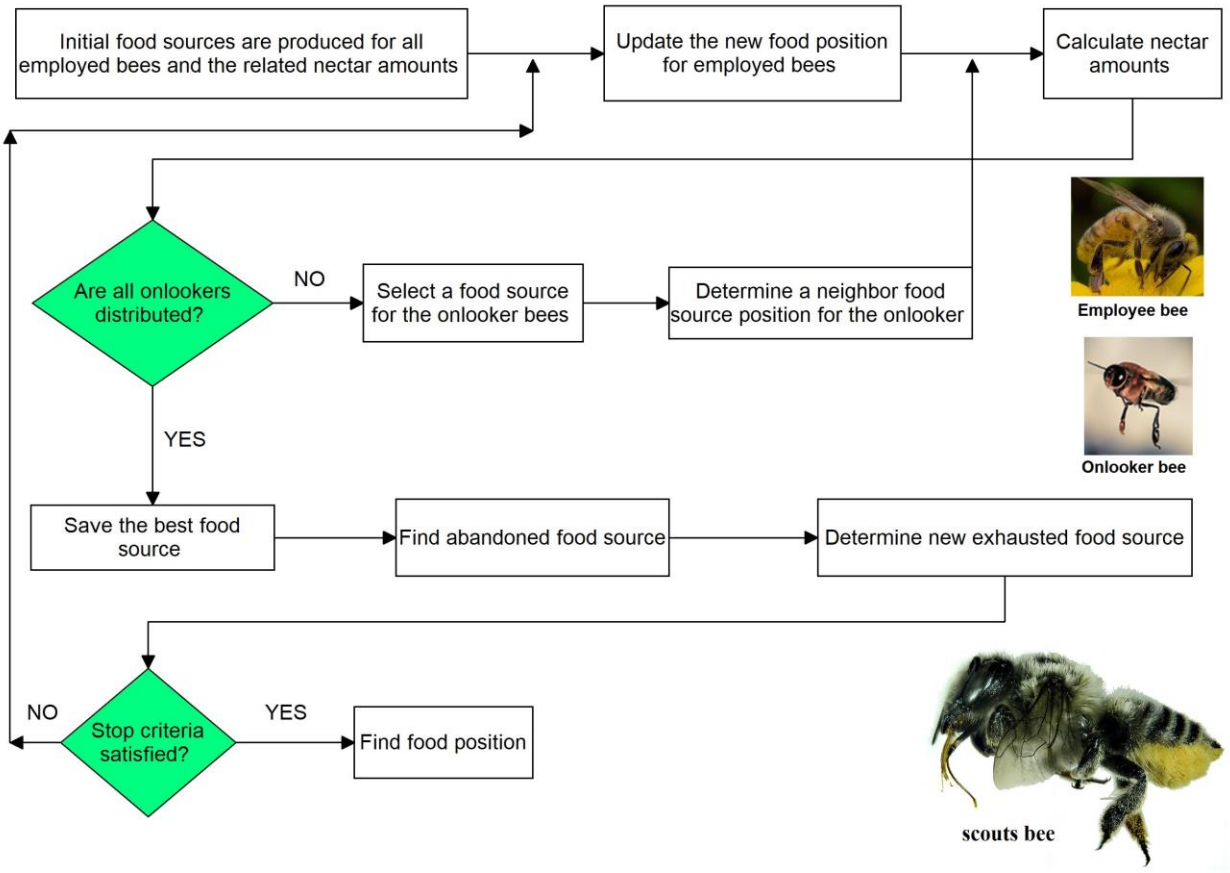

Figure 1. The ABC evolutionary algorithm procedure flowchart.

2.3 Support vectors

The concept of "support vector machine" (SVM) was first mentioned by Vapnik in 1995, and in time it has become one of the most well-known and commonly used optimal technique for data classification (Vapnik, 1995) due to its good mathematical formulation accompanied by numerous empirical results (Czarnecki & Tabor, 2014). For more thorough descriptions, the interested reader is referred to the studies by Burges (1998), Theodoridis and Koutroumbas (2003), Hsu, Chang, and Lin (2003).

In data mining and classification, support vectors are the samples which separate classes or categories from each other based on labels and they are mostly used in supervised learning. In other words, support vectors are the border line between categories and finding them helps in identifying outliers and less desirable samples. To decrease computation time and ease the computation process, the outliers will be eliminated for the rest of the process, and after performing the classification based on the remaining data (support vectors), they will return just for the end user analysis. So, outliers are removed temporarily just for the purposes of finding support vectors and for the classification process, but data will be intact at the end of the entire process. Figure 2 presents the position of the support vectors in two dimensions for three classes, which should be found. Samples with black circles around them are support vectors.

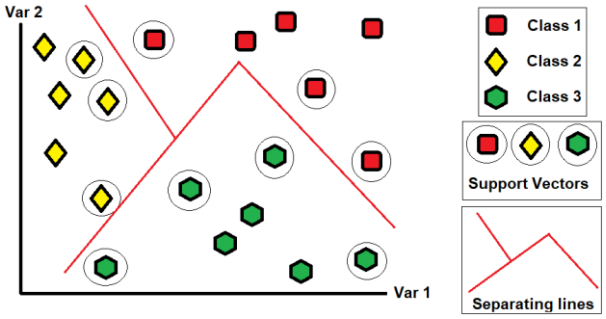

Figure 2. The position of the support vectors in the data.

3. Literature review

One of the best and most powerful support vector finder method was proposed by Cortes and Vapnik in 1995 (Cortes & Vapnik, 1995) under the title of "support vector network"; this has become a powerful tool in many fields, such as classification, pattern recognition, detection, gene selection, and so on (Qi, Yang, Hu, & Yang, 2019). In time, many research studies have been carried out based on it; also, efforts have been made to develop variants of it. Nowadays, the method is known under the name of "Support Vector Machine" (SVM), and it is still a very powerful classification tool. Despite this, however, SVM is bit slow in the test phase. Below we proceed to offer an overview of the existing literature on the topic.

In 1996, Burges (1996) proposed a method to compute an approximation to the decision rule in terms of a reduced set of vectors. These reduced set vectors are not support vectors and can in some cases be computed analytically. Burges actually decreased the computation complexity for SVM with no loss in the generalization performance. Another noteworthy research study dedicated to decreasing the SVM runtime is that by Burges and Schölkopf, published in 1996 (Burges & Schölkopf, 1996). On the one hand, the authors improved the accuracy by incorporating knowledge about invariances of the problem at hand, and on the other hand, they improved the classification speed by reducing the complexity of the decision function representation.

In 1999, Syed, Huan, Kah, and Sung (1999) presented an approach for incremental learning with Support Vector Machines. Their approach could effectively deal with the changes in the target concept that are the result of the incremental learning setting. In the same year, Platt (1999) further proposed a method to change the SVM learning process, called Sequential Minimal Optimization, or SMO. In fact, he broke down the SVM Quadratic Programming (QP) optimizations into series of smallest possible QP problems using the SMO.

In 2000, Schölkopf, Smola, Williamson, and Bartlett (2000) proposed a new class of support vector algorithms for regression and classification, using different parameters to control the number of support vectors, based on research carried out on existing methods. In 2001, Downs, Gates, and Masters (2001) proposed statistical changes in the SVM structure as follows. They proposed an algorithm to detect and eliminate the unnecessary linearly dependent support vectors, and they tested their method on some benchmark data sets for classification and regression task using different kernels and different parameters.

Another noteworthy research study in this area belongs to Keerthi, Chapelle, and DeCoste (2006), who in 2006 introduced a method to improve the speed of the SVM in big data processing. The method works as follows: First, it separates the idea of basis functions from the concept of support vectors. Second, the

system greedily finds a set of kernel basis functions of a specified maximum size to approximate the SVM primal cost function well; and in their system, the number of basis functions required to achieve an accuracy close to the SVM accuracy is usually far less than the number of SVM support vectors. Another research study that decreases the SVM speed belongs to Nguyen and Ho (2006). Their method consisted in simply replacing a new point for each two points or samples from the same class.

More recently, in 2017, Mousavi, MiriNezhad, and Mirmoini (2017) proposed a system to find support vectors based on K-means clustering and triangular calculations and tested their method with least squares and SVM classification algorithms on Fisher's Iris (Fisher, 1936) and Wine data sets (Aeberhard, Coomans, & De Vel, 1992).

Another method to remove unnecessary data and speed up the calculation was proposed by Mirinezhad, Dezfoulian, Mosleh, and Mousavi (2017). The authors used the Multi Class Instance Selection (MCIS) algorithm by Chen, Zhang, Xue, and Liu (2013) to obtain the most valuable data from each class and speed up the Widrow-Hoff classification algorithm by Steinbuch and Widrow (1965). A similar method is presented in Dezfoulian, MiriNezhad, Mousavi, Mosleh, and Shalchi (2016). In this case, the authors used MCIS in the first step and the Ho-Kashyap algorithm by Ho and Kashyap (1965) in the classification step.

Each of the above-mentioned methods have their pros and cons. Some of them are good in dealing with big data and some other in dealing with specific types of data sets. In a nutshell, Cortes and Vapnic (1995) introduced the traditional form of SVM and Burges (1996) decreased the computation complexity of the same. Burges and Schölkopf (1996) further improved the accuracy of SVM. Platt (1999) changed the learning process of SVM with SMO. Downs, Gates, and Masters (2001) proposed an algorithm to detect and eliminate the unnecessary linearly dependent support vectors. Also, Keerthi, Chapelle, and DeCoste (2006) improved the speed of SVM. Mousavi, MiriNezhad, and Mirmoini (2017) proposed a system using K-means clustering and triangular calculations to find support vectors. Mirinezhad, Dezfoulian, Mosleh, and Mousavi (2016) combined MCIS clustering and Widrow-Hoff classification algorithms to obtain the closest samples from each class and speed up the classification process. Table 1 provides a summary of the strengths and weaknesses of our proposed approach versus these other approaches.

Table 1. Proposed approach versus existing approaches.

Studies	**Strengths**	**Weaknesses**
Cortes and Vapnik (1995)	Traditional form of SVM.	SVM is slow in the test phase.
Burges and Schölkopf (1996)	Improved the accuracy of SVM.	Speed like SVM.
Platt (1999)	Breaks down the SVM Quadratic Programming (QP) optimisations into a series of smallest possible QP problems using the Sequential Minimal Optimisation (SMO).	Good only for QP problems.
Downs, Gates, and Masters (2001)	Proposed an algorithm that detects and eliminates the unnecessary linearly dependent support vectors.	Low speed.
Keerthi, Chapelle, and DeCoste	Improved the speed of SVM.	Less accuracy.

(2006)		
Dezfoulian, MiriNezhad, Mousavi, Mosleh, and Shalchi (2016).	Combined MCIS clustering and Widrow-Hoff classification algorithms to obtain the closest samples from each class and speed up the classification process.	Very low accuracy.
Mousavi, MiriNezhad, and Mirmoini (2017)	Proposed a system to find support vectors based on K-means clustering and triangular calculations.	Medium speed and accuracy.
Proposed Approach: PSV Finder Method	Nature-inspired outlier removal system. Good accuracy and speed in most data sets.	When the number of samples increases, the approach performs slower than traditional methods (but still, with good accuracy).

4. Proposed evolutionary pentagon support vector finder method

In order to find support vectors, the first phase is to reduce the data using evolutionary clustering, which is performed by means of the proposed modified Artificial Bee Colony together with the Fuzzy C Means-Manhattan method (ABC+FCMM). The second phase is to classify and label the remaining clusters based on the previous labels. The third phase is to remove the outliers based on a threshold α. The last phase involves calculating the pentagon area and the angle for each i^{th} current sample from the j^{th} class and the 4 samples from the other k^{th} class. In this last phase, if the area and angle thresholds (η and θ) are satisfied, then the sample from the i^{th} class is a support vector, otherwise it is not. All these support vectors will further be used in the classification process. Figure 3 shows the flowchart of the proposed evolutionary pentagon support vector finder method. The proposed method is called evolutionary Pentagon Support Vector finder or PSV.

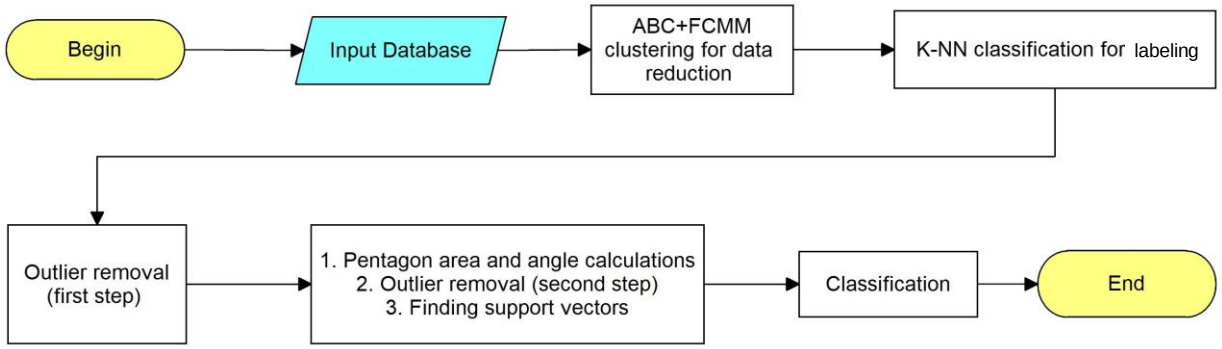

Figure 3. Proposed Evolutionary Pentagon Support Vector (PSV) Finder flowchart.

4.1 Artificial Bee Colony (ABC) + Fuzzy C Means-Manhattan (FCMM) clustering

Clustering and classification are two powerful tools in data mining for dividing data into categories. The difference between them consists in whether the data have a predefined label or not. If group labels are known beforehand, then we are dealing with supervised learning and the problem is a classification

problem, and if group labels are not predefined, then we are dealing with unsupervised learning and the problem is a clustering or automatic classification problem that needs to be solved. Now, if in clustering the number of clusters is unknown, then it is called automatic clustering. The goal of clustering here is to reduce the data for further processing purposes, which needs heavy computations, especially when dealing with big data.

K-means or Lloyd's clustering algorithm (Lloyd, 1982) is a metaheuristic clustering method with high calculation speed, but with an important shortcoming. The efficiency and performance of the K-means algorithm is greatly affected by the initial cluster centers, as different initial cluster centers often lead to different clustering. Therefore, the problem with using the K-means lies in determining the initial cluster centers, which are random and can lead to very poor and unstable clustering results. Fuzzy C-means (FCM) (Dunn, 1973) is the fuzzy model of the K-means and suffers from a similar issue. In non-fuzzy or hard clustering, data is divided into distinct clusters, wherein each data point can only belong to exactly one cluster. In fuzzy clustering, on the other hand, the data points can potentially belong to multiple clusters. It is possible to fix the problem of the initial cluster center in hard clustering or fuzzy clustering by solving it using intelligent evolutionary algorithms, such as Genetic Algorithm (Mitchell, 1998), Harmony Search (Geem, Kim, & Loganathan, 2001), Artificial Bee Colony (ABC) (Karaboga, 2005), among others.

In this paper, we combine the ABC EA with the FCM clustering algorithm so as to determine the proper initial centers of the clusters, based on the paper by Karaboga and Ozturk (2010) and with a small change in distance calculation. We used city block or Manhattan distance instead of the Euclidean distance, which leads to better results. Also, the paper by Karaboga and Ozturk (2010) employed ABC + Fuzzy Clustering (FC), whereas here we used ABC+FCM. So, our modified clustering method is called Artificial Bee Colony + Fuzzy C Means with Manhattan distance clustering or ABC+FCMM clustering. FCM clustering algorithms gain the membership values between 0 and 1, which indicates the degree of membership of each sample to each cluster. The proposed ABC+FCMM clustering problem is defined as follows:

$$\mu_{ij}\,[0,1], i = 1,2,\ldots.n\,; j = 1,2,\ldots n \tag{3}$$

$$\sum_{j=1}^{c} \mu_{ij} = 1 \quad i = 1,2,\ldots,n \tag{4}$$

$$0 < \sum_{i=1}^{n} \mu_{ij} < n \quad j = 1,2,\ldots c \tag{5}$$

Here, μ is a fuzzy matrix with n rows and c columns, wherein furthermore, n is the number of samples and c is the number of clusters. μ_{ij} is the element in the i^{th} row and j^{th} column in μ and represents the degree of membership function of the i^{th} sample to the j^{th} cluster. Also, the FCMM objective function is to minimize the following expression:

$$\sum_{j=1}^{c}\sum_{i=1}^{n} \mu_{ij}^{m} ||x_i - c_j|| \tag{6}$$

where:

(7)

$$\mu_{ij}^{m} = \frac{1}{\sum_{k=1}^{c}\left(\frac{\left||x_i - c_j|\right|}{\left||x_i - c_k|\right|}\right)m - 1}$$

Here, x is the number of samples, c is the number of clusters, and m is the related fuzzy index. The equation is changed based on the Manhattan distance. Here, we imagine bees as samples and the strongest bee as the cluster center during cycles so as to determine the best initial cluster center. Table 2 shows the pseudo code for the proposed ABC+FCMM clustering method.

Table 2. Proposed ABC+FCMM pseudo code.

Generating the initial population X_i, i=1…SN from EQ. (2)
Evaluating the population
Cycle=1
Do
For each employee bee
Produce new solution V_i from EQ. (1)
Calculating the fitness function value
Greedy selection (first)
For each onlooker bee
Choose a solution
Produce new solution V_i
Calculating the fitness function value
Greedy selection (second)
If there is an abandoned solution,
Then
Replace the solution with a new random one
Scout solution using EQ. (2)
Memorizing the best solution ever yet
Cycle = Cycle + 1
Until cycle = Maximum Cycle Number (MCN) (***SOMETIMES IT TAKES 2000 CYCLES FOR CLUSTERS TO SET***)

Figure 4 shows 2 classes of samples (random) in 2 dimensions, with 14 samples per each class, which are clustered using the proposed ABC+FCMM clustering method into 5 clusters for each class.

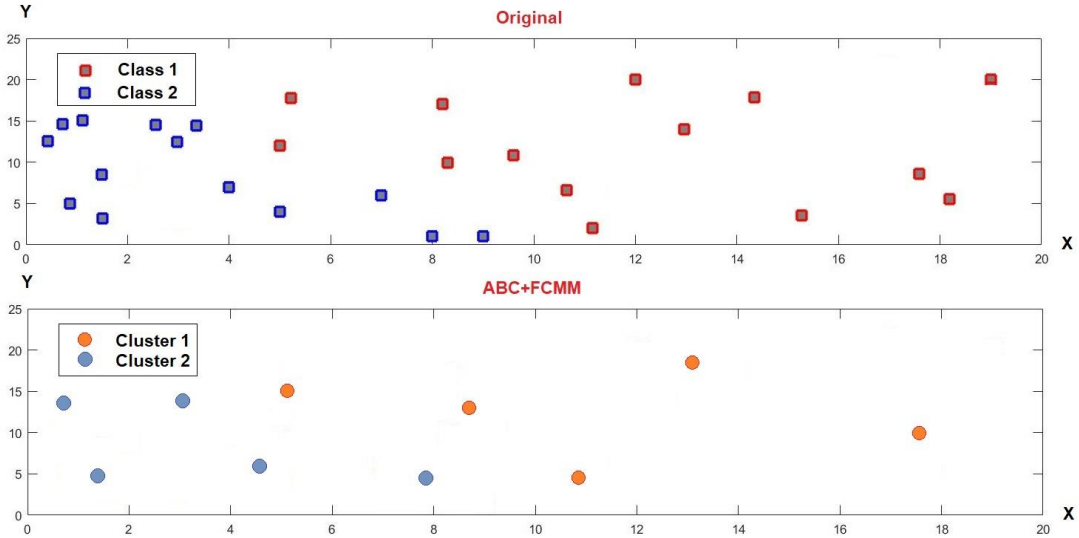

Figure 4. An example of application of the proposed modified ABC+FCMM clustering method on random data.

4.2 K-Nearest Neighbourhood (K-NN)

After reducing the data using the proposed modified ABC+FCMM clustering method, it is time to label the clusters using previous data labels and assign them to categories. To this aim, the K-Nearest Neighbourhood (K-NN) classifier (Altman, 1992) is employed. Due to its fast calculations and for the present purposes, using K-NN here is enough. Imagine that there are three classes of samples in four dimensions, and each class has 100 samples. Suppose we use ABC+FCMM with 20 clusters for each class, which will change the data from 300 samples to 60. At this stage, data are unlabeled, which is not desirable for further support vector finding and final classification. So, all 60 samples should be assigned to the closest class or category, using previous data labels, which are saved for this stage. For that, and for each sample, a window with K size (in diameter) in the space will be determined, and the class that has a higher number of samples in the window will claim the current sample. Then, the clusters will be labeled as classes just like in the previous clustering process, but with fewer data points. As it is clear, K must be an odd number and this threshold is different depending on each data set; but the lower the threshold, the better the classification result. K-NN is a simple classification algorithm and we did not change the structure of it, but instead used the original structure. It starts with K (odd number) window size and calculates the Euclidean distance between the current sample and the samples inside the window, and further assigns the sample to the class with more samples inside the window. For more information about this algorithm, the interested reader is referred to the study by Altman (1992). Here, we first train the data with the original data set and then use the clusters as the test data. Figure 5 exemplifies the use of the K-NN classifier with k = 3 to classify each cluster into the closest samples. Other clusters calculate the same as the example.

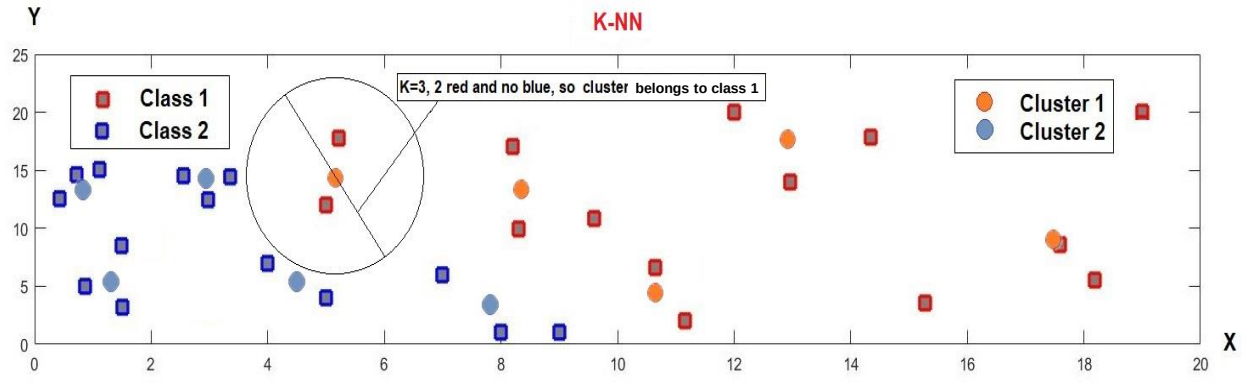

Figure 5. Using the K-NN classifier with k = 3 to classify each cluster into the closest samples.

4.3 First step in outlier removal

After labelling the clusters, it is time to eliminate some of the far outlier data. For that, a Parzen window (Parzen, 1962) just like K-NN will be used. If there are no samples beyond the threshold α and also there are no samples from other classes, then that sample is an outlier and must be removed. The value of the threshold α varies and it depends on the upper and lower bound values of the variables in the data set. Figure 6 exemplifies this concept for two samples. In light of the above comments, one sample is an outlier and the other one is not. Also, this process further takes place for all the samples.

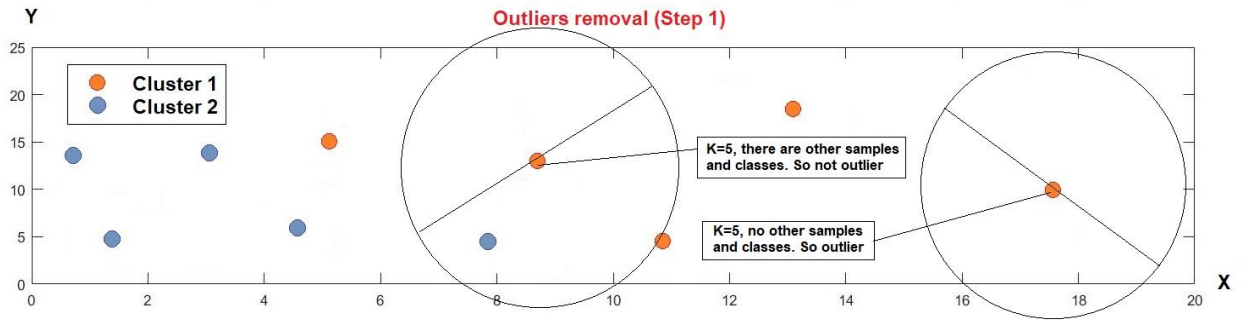

Figure 6. An example of application of the proposed first step in outlier removal on random data.

4.4 Pentagon area and angle

After removing the outlier samples in the first step, it is time to calculate the position of the final support vectors. To that effect, the pentagon area and angle are considered, which are based on 1 sample from the current class and 4 samples from the other classes. The regular pentagon area is calculated as 5/2 * (size of one side * apothem length), as can be seen in Figure 7, but it is not always like that. In fact, it is not like that in 99% of the calculations, wherein we need to calculate the irregular polygon (here pentagon shape) area instead. To calculate the irregular polygon area, the following equation (8) is used, when the coordinates are known.

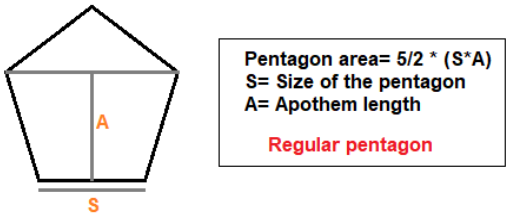

Figure 7. Regular pentagon area calculation.

$$Irrigular\ polygone\ (pentagon)\ area = \left|\frac{(x_1y_2 - y_1x_2) + (x_2y_3 - y_2x_3) \dots\dots + (x_ny_1 - y_nx_1)}{2}\right| \quad (8)$$

where x_1 is the x coordinate of vertex 1 and y_n is the y coordinate of the n^{th} vertex and so on (see Figure 8a). Also, the sum of the inside angles of a pentagon is equal to 540 degrees or equal to three triangles' degrees, which is 180 + 180 + 180, as can be observed from Figure 8b. There is a general rule of (n-2)*180 to calculate the sum of the angles for any type of polygon, such as pentagons, hexagons, octagons, and so on, wherein n is the number of vertices. For example, this value for the pentagon is (5-2)*180 = 540 degrees and for the octagon is (8-2)*180 = 1080 degrees. Also, for calculating the interior angles of the irregular polygon, Matlab software is employed.

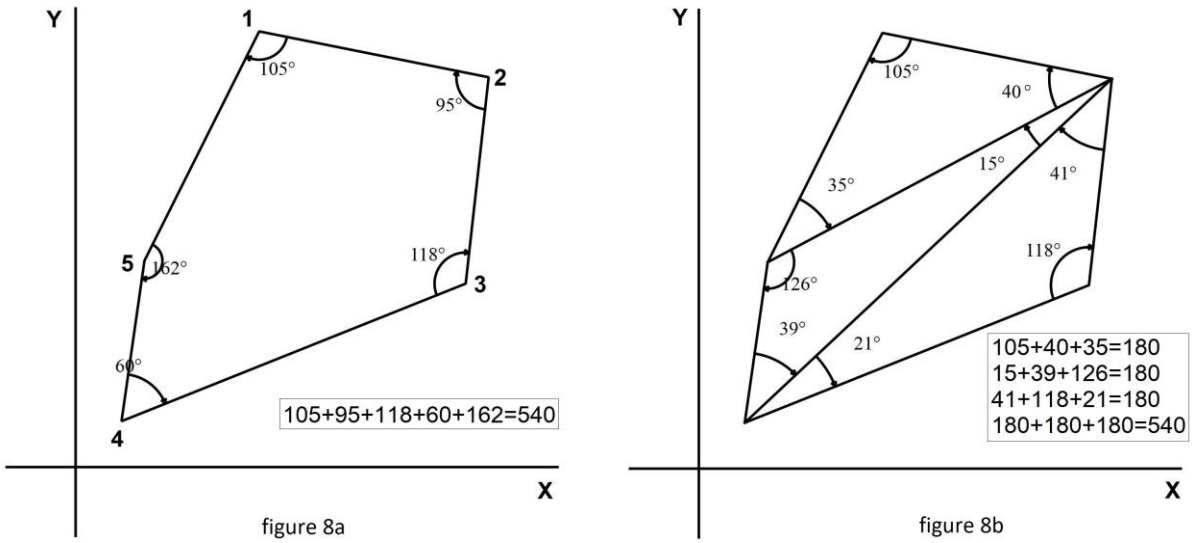

Figure 8. Pentagon sum of angles (left, figure 8a), Pentagon division into three triangles (right, figure 8b).

So, there are two conditions to satisfy here. First, the area value of the polygon should not be more than the threshold η and each angle degree should be less than the threshold θ, as Figure 9 shows. If both conditions are satisfied, then that sample is a support vector; if not, then it is an outlier sample and must be eliminated. Figure 9 shows two examples of conditions not being satisfied for area and angle. Now, the remaining data which are support vectors will be used in the classification task.

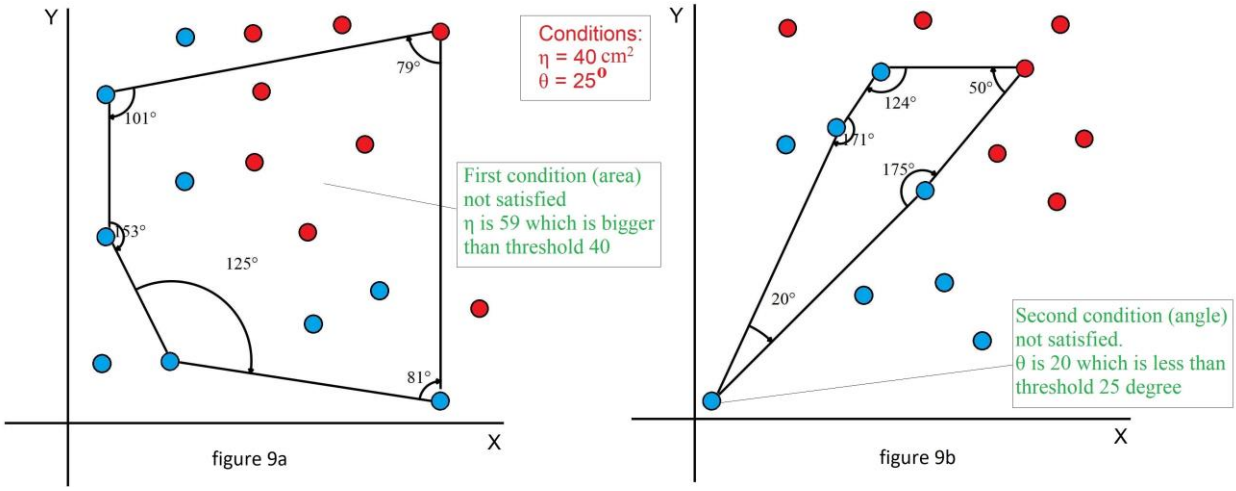

Figure 9. Two examples of conditions not being satisfied for area (left, figure 9a) and angle (right, figure 9b)

Figure 10 shows the application of the proposed support vector finder procedure, step by step and in visual form, to Fisher's Iris data set (Fisher, 1936) (variables 3-4, classes 2-3). As it is clear from Figures 10g and 10h, the proposed method using the simple least squares classification works just as well as the SVM classification with 5 misclassified samples.

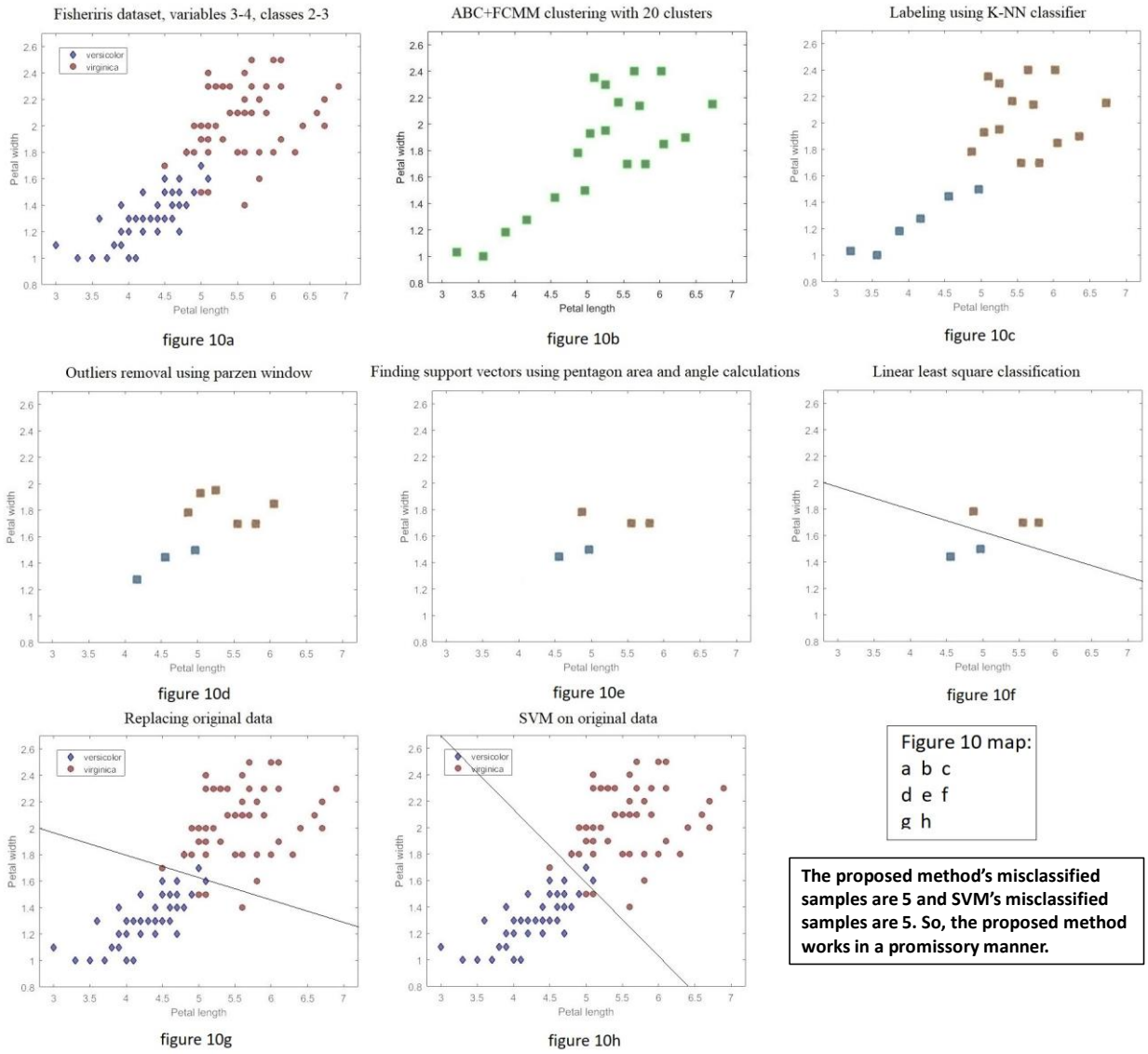

Figure 10. The proposed support vector finder procedure, step by step, in visual form, applied to Fisher's Iris data set (variables 3-4, classes 2-3).

5. Validations and results

As it is clear from applying our method, after finding the support vectors, the position of the samples is changed due to the clustering process in the first step. So, after the classification task, the acquired separating lines or hyper planes will be used on the real, unchanged data to determine the error percentage based on the positions of the real samples. Validations are performed on 7 different classification benchmark data sets. As validation is going to be performed with classification algorithms, data should be labeled and supervised, and clustering and regression data sets are useless here. Data sets used in this research are Fisher's Iris (Fisher, 1936), User Knowledge Modelling (Kahraman, Sagiroglu, & Colak, 2013), Blood Transfusion Service Center (Yeh, Yang, & Ting, 2009), Haberman's Survival (Haberman, 1976), Wine (Aeberhard, Coomans, & De Vel, 1992), Ionosphere (Sigillito, Wing, Hutton, & Baker, 1989), and EEG Eye State (Rösler & Suendermann, 2013).

Also, we use the SVM (Cortes & Vapnik, 1995), K-NN (Altman, 1992), and Artificial Neural Network (LeCun, Bengio, & Hinton, 2015) classifiers on the original data in these data sets for classification purposes; furthermore, we also use them after applying the proposed method on the original data in order to compare the results obtained and evaluate the performance of the proposed method. As mentioned before, the proposed method is called *evolutionary Pentagon Support Vector finder* or *PSV*.

5.1 Data sets

As mentioned before, seven classification data sets are employed for the evaluation of the proposed method. The information of the data sets is described in detail in Table 3. We have tried to incorporate data sets with different number of attributes, classes, and instances, so as to cover all types of numeric classification circumstances, for a more robust validation.

Table 3. The seven data sets used in our evaluation.

Name	Fisher's Iris	EEG Eye State	Wine	Haberman 's Survival	User Knowledge Modelling	Ionosphere	Blood Transfusion
Reference	Fisher (1936)	Rösler and Suendermann (2013)	Forina *et al.* (1988)	Haberman (1976)	Kahraman, Sagiroglu, and Colak (2013)	Sigillito *et al.* (1989)	Yeh, Yang, and Ting (2009)
Characteristics	Multivari ate	Multivariate, Sequential, Time-Series	Multivaria te	Multivaria te	Multivariate	Multivariate	Multivariate
Number of Instances	150	14980	178	306	403	351	748
Number of Attributes	4	15	13	3	5	34	5
Attribute Characteristics	Real	Integer, Real	Integer, Real	Integer	Integer	Integer, Real	Real
Number of Categories	3	2	3	2	4	2	2
Associated Task(s)	Classifica tion	Classification	Classificat ion	Classificat ion	Classification, Clustering	Classification	Classification
Area	Life	Life	Physical	Life	Computer	Physical	Business
Year of Creation	1936	2013	1990	1976	2009	1989	2008
Number of Citations until 2018	14442	29	135	102	52	297	168

5.2 Classifiers

For validation purposes of the proposed method, the three classifiers mentioned above are used on the original data, so as to see under which circumstances (based on the structure of the data sets) the proposed method is efficient or not when compared to the classification performed by the three classifiers on the original data. The classifiers used are K-NN (Altman, 1992), SVM (Cortes & Vapnik, 1995), and Artificial NN (LeCun, Bengio, & Hinton, 2015). K-NN is working based on the samples (neighbours) closest to each other and the number of K nearest neighbours in the space. Also, this algorithm is an instance-based learning algorithm and one of the simplest algorithms in machine learning. SVM, on the other hand, is one of the most famous algorithms in machine learning and pattern recognition for different tasks. It is possible to use this network-type algorithm for more than just a classification task, for example, for regression, in which case it is called Support Vector Regression (SVR); in clustering, it is called Support Vector Clustering (SVC). Finally, the Artificial Neural Network is the computer version of the brain structure and it is inspired by nature. All the learning structure in NN (LeCun, Bengio, & Hinton, 2015) is based on transmitting data between three interconnected layers: input, hidden (which may include more than one layer), and output. We are using them in the classification task here.

5.3 Classification results

Tables 4, 5, and 6 show the classification and runtime speed results for the seven data sets, with different kernels and parameters, respectively, with classifiers applied on both original data and on data on which we have first applied our proposed PSV finder method. As it can be observed from these tables, in some cases, the proposed method performs better when compared to existing methods. The runtime speed of the proposed method in all tests is higher than the runtime speed of tests on original data, which shows higher computation in our approach. The SVs in the fourth column of the tables indicate the number of support vectors that remain after applying PSV. The support vectors that remain are used for the classification task with classification algorithms. For example, in the EEG Eye State data set and for classification using SVM, there is 4.68% accuracy improvement by using the proposed method. Nonetheless, for Haberman's data set and for classification using Artificial NN, there is -2.29% weakness in our method when compared to using Artificial NN on the original data. In summary, there are both improvements and weaknesses in using the proposed PSV finder method on different data sets with different classifiers, with estimation of ± 3% recognition accuracy, which shows the good performance of our method in dealing with different conditions, such as the number of instances and attributes.

Table 4. SVM on seven data sets using different kernels.

Data set	Parameter	On original data-runtime	Selected SVs	After PSV-runtime
Fisher's Iris	Cubic	94.4% ± 1 % - 0.2 s	39	95.7% ± 1% - 0.48 s
EEG Eye State	Fine Gaussian	81.3% - 23.14 s	2588	85.98% - 32.81 s
Wine	Coarse Gaussian	98.9% - 0.73 s	12	98.71% - 2.45 s
Haberman's Survival	Quadratic	74.4% ± 1 % - 4.11 s	40	77.37% ± 2% - 7.12 s
User Knowledge Modelling	Quadratic	94.6% - 1.20 s	76	94.0% ± 2% - 1.81 s
Ionosphere	Medium Gaussian	95.9% ± 2 % - 1.18 s	52	97% - 3.97 s
Blood Transfusion	Fine Gaussian	78.2% - 0.72 s	112	81.27% - 0.96 s

Table 5. K-NN on seven data sets using different kernels.

Data set	Parameter	On original data	Selected SVs	After PSV
Fisher's Iris	Fine	94.7% ± 1 % - 1.3 s	39	93.7% ± 1% - 3.5 s
EEG Eye State	Weighted and Ensemble subspace	86.1% ± 2 % - 5.14 s 96.6% - 12.27 s	2588	80.12% ± 3% - 9.9 s 95.3% ± 1% - 16.81 s
Wine	Weighted	96.2% - 0.68 s	12	97.8% - 3.3 s
Haberman's Survival	Coarse	73.5% - 0.73 s	40	75.61% - 2.3 s
User Knowledge Modelling	Ensemble subspace	85.6% ± 1% - 2.19 s	76	87.6% ± 1% - 3.26 s
Ionosphere	Ensemble subspace	92.3% -2.83 s	52	91.5% - 6.33 s
Blood Transfusion	Cubic	77.7% - 1.22 s	112	75.0% - 1.57 s

Table 6. Artificial NN on seven data sets using different parameters.

Data set	Parameters	On original data	Selected SVs	After PSV
Fisher's Iris	Sigmoid hidden layers (10) - Soft-max output Back-propagation learning	98.2% ± 1 % - 0.1 s	39	97.8% ± 1% - 0.47 s
EEG Eye State	Sigmoid hidden layers (1000) - Soft-max output Back-propagation learning	62.42% - 5.52 s	2588	60.8% - 11.28 s
Wine	Sigmoid hidden layers (5) - Soft-max output Back-propagation learning	100% - 0.35 s	12	100% - 0.94 s
Haberman's Survival	Sigmoid hidden layers (100) - Soft-max output Back-propagation learning	84.11% - 0.1 s	40	81.82% - 0.3 s
User Knowledge Modelling	Sigmoid hidden layers (20) - Soft-max output Back-propagation learning	96.2% - 0.25 s	76	97.0% - 0.69 s
Ionosphere	Sigmoid hidden layers (2) - Soft-max output Back-propagation learning	94.7% ± 3 % - 0.31 s	52	96.1% ± 1% - 0.68 s
Blood Transfusion	Sigmoid hidden layers (14) - Soft-max output Back-propagation learning	83.30% - 0.5 s	112	80.4% - 0.91 s

Figure 11 shows the Receiver Operating Characteristic (ROC) curve (Hanley & McNeil, 1982) for SVM after applying the proposed PSV finder method on Fisher's Iris and for each class. The ROC curve is created by plotting the true positive rate (recall or probability of detection) against the false positive rate (fall-out or probability of false alarm) at various threshold settings. Also, Figure 12 represents the confusion matrix for the User Knowledge Modelling and Ionosphere data sets, from left to right, using the K-NN classifier after applying the proposed PSV finder method. The confusion matrix shows the percentage of classification for each class in the diagonal cells in the table, and the misclassification percentage in relation to each class is shown in the rows of the respective classes. Figure 13 shows the Blood transfusion data set's Performance, Training state, Error histogram, and Test confusion matrix plot using the Artificial NN classifier. Finally, Figure 14 presents the proposed PSV finder method's error percentage for the seven data sets using the SVM, K-NN, and Artificial NN classifiers.

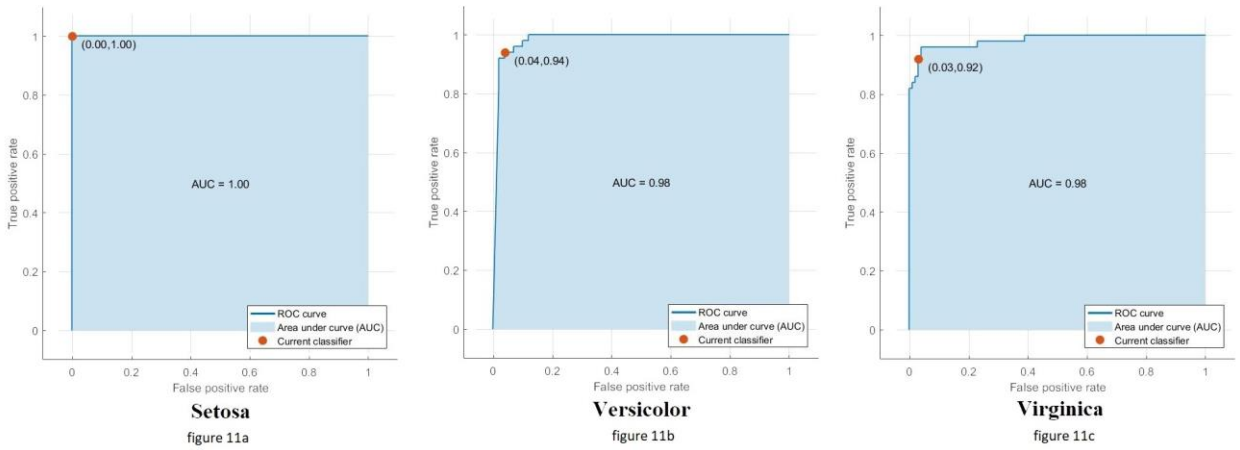

Figure 11. The Receiver Operating Characteristic (ROC) curve for SVM after applying the proposed PSV finder method on Fisher's Iris for each class.

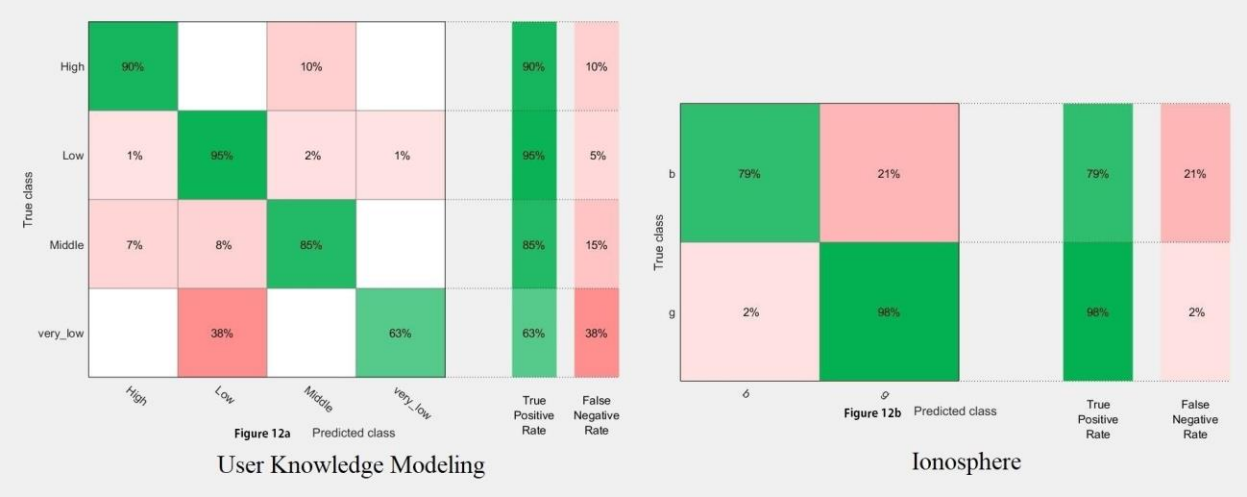

Figure 12. Confusion matrix for User Knowledge Modelling and Ionosphere data sets from left to right using the K-NN classifier after applying the proposed PSV finder method.

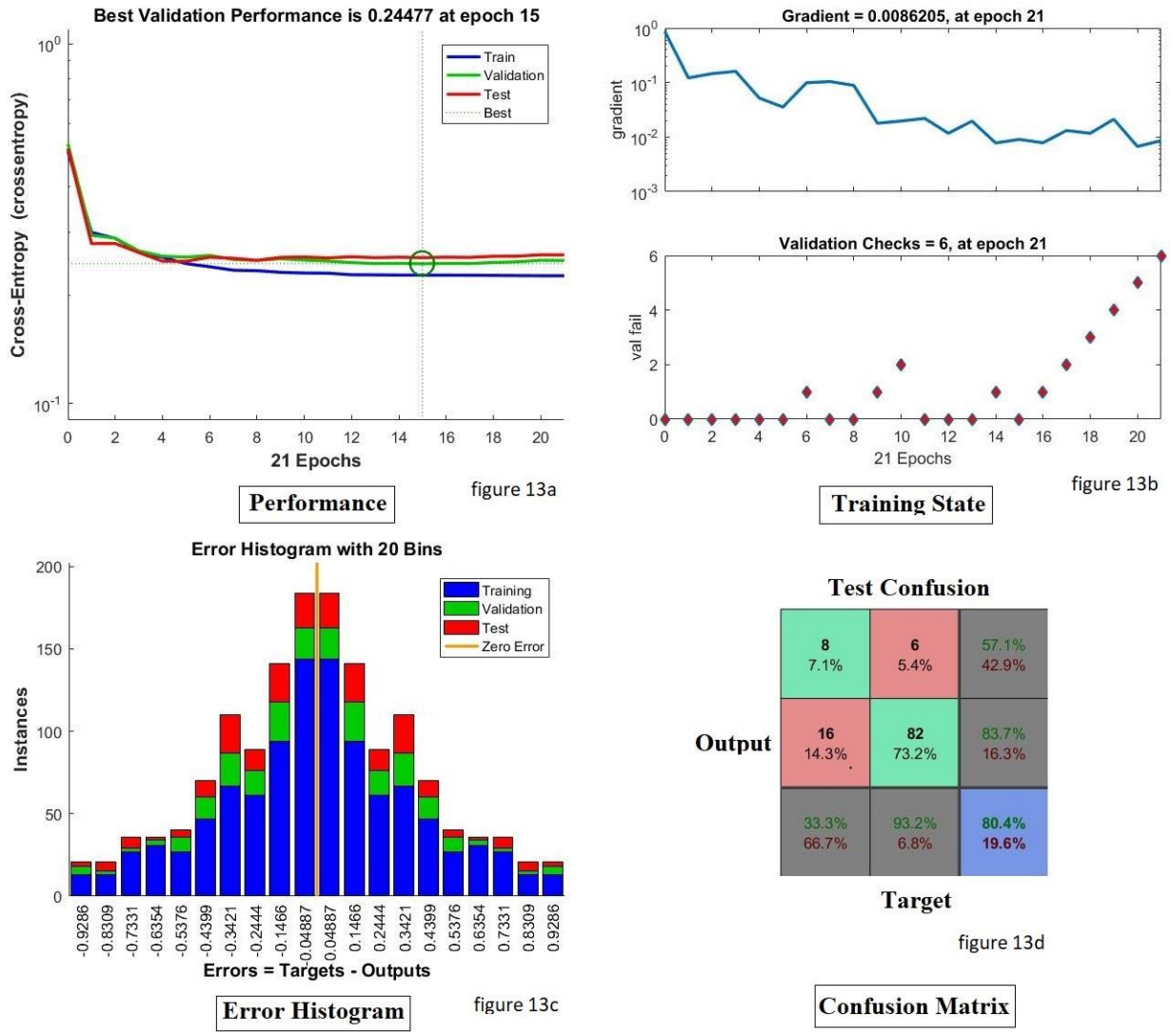

Figure 13. Blood transfusion data set's Performance, Training state, Error histogram and Test confusion matrix plot using the NN classifier.

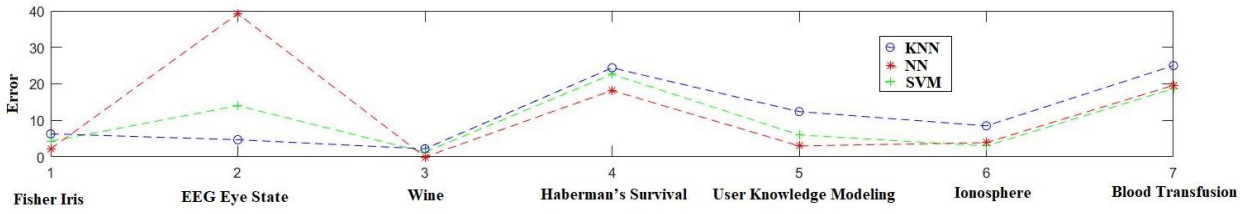

Figure 14. The proposed PSV finder method's error percentage for the seven data sets using the SVM, K-NN, and NN classifiers.

6. Conclusions

In the present paper, we proposed a method called evolutionary Pentagon Support Vector finder (PSV) for finding support vectors based on evolutionary clustering and pentagon geometrical computations. The algorithm starts with the Artificial Bee Colony evolutionary clustering for locating the position of the initial cluster centers, while also reducing a significant number of data points for both normal and big data analytics. Each cluster is representative of the closest samples' class based on the K-Nearest Neighbourhood classification algorithm and they will be labelled based on that to change the cluster samples (unsupervised) to being classified (supervised). To remove outliers (first step in outlier removal), we apply a method based on variable-size window. After reducing the data and removing the outliers, we calculate the pentagon area and angle to determine the final support vectors for each class. The selection of samples (support vectors) is based on calculating the pentagon area and angle for each one sample from the present class and four samples from the other classes, which gives five samples for pentagon calculations. If the pentagon area and the angle are in the threshold range, then that sample is recognized as a support vector; otherwise, the algorithm proceeds with the second step in outlier removal. At the end and for validation purposes, we performed few classifications with K-NN, Support Vector Machine (SVM), and Artificial Neural Network classifiers to separate the classes' support vectors. Finally, the method was tested with benchmark data sets and compared with classification algorithms applied on both original data and on data on which we have first applied our proposed PSV finder method.

The results returned are promising. Findings show that using evolutionary clustering could help not only to reduce the number of data points in big data, but it could also fix the problem of setting the initial cluster centers that is present in many of the traditional clustering methods. Also, in this approach, final accuracy is not only unharmed, but it also increases in the case of some data sets. In terms of limitations, it is to be noted that when the number of samples increases, our method will perform slightly slower than traditional methods; despite this, however, the accuracy is still good and, as mentioned, it even improves in some cases.

For future research, it would thus be interesting to use other evolutionary clustering methods, such as the Imperialist Competitive Algorithm (ICA) (Atashpaz-Gargari & Lucas, 2007), BAT algorithm (Yang, 2010), Galaxy Gravity Optimization (GGO) (Mousavi, MiriNezhad, & Dezfoulian, 2017), or even Harmony search (Geem, Kim, & Loganathan, 2001) to set the initial cluster centers and compare the results. Changing the pairwise distance method could further be helpful. Also, changing the geometrical shape to a hexagon or octagon to find final support vectors could be considered, although it will increase runtime speed and will need to eliminate more outliers in the first step of the outlier removal process. We hope that these types of combined methods and computations will open avenues to find better and more robust methods in machine learning. With better technology, solving complex problems gets easier in terms of computation speed, but accuracy is still a challenging area, which nonetheless is possible to tackle using such combined methods. Our paper represents a contribution to this strand of research.

From a methodological perspective, our proposed method is superior when compared to existing approaches in view of the use of geometrical calculations and evolutionary algorithms to make a more effective system. The contribution of the present paper resides in the introduction of a unique approach to eliminating outliers, which consists of four steps: (1) reducing the amount of data using evolutionary clustering (Artificial Bee Colony together with the Fuzzy C Means-Manhattan method), (2) labelling the remaining data using the K-nearest neighbourhood classifier, (3) removing outliers based on a specific

threshold, and (4) calculating the area of the pentagon and the angle between samples of existing classes to determine the position of the final support vectors.

From a practical perspective, the present work is important because we were able to validate the system on real life acquired data sets, such as Fisher's Iris, EEG Eye State, Wine, Haberman's Survival, User Knowledge Modelling, Ionosphere, and Blood Transfusion; hence, the proposed approach could be extended to practical problems.

Acknowledgement:
The authors would like to thank the editor in chief and the anonymous reviewers for very helpful comments on the previous draft of this manuscript.

References

Aeberhard, S., Coomans, D., & De Vel, O. (1992). Comparison of classifiers in high dimensional settings. *Techical Report 92-02*. Department of Computer Science and Department of Mathematics and Statististics, James Cook University of North Queensland, Australia.

Akbari, R., Hedayatzadeh, R., Ziarati, K., & Hassanizadeh, B. (2012). A multi-objective artificial bee colony algorithm. *Swarm and Evolutionary Computation, 2*, 39-52.

Alatas, B. (2010). Chaotic bee colony algorithms for global numerical optimization. *Expert Systems with Applications, 37*(8), 5682-5687.

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician, 46*(3), 175-185.

Atashpaz-Gargari, E., & Lucas, C. (2007). Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. *IEEE Congress on Evolutionary Computation (CEC 2007)*, 4661-4667.

Bäck, T., Fogel, D. B., & Michalewicz, Z. (1997). *Handbook of evolutionary computation.* New York, NY: Oxford University Press; Bristol, UK: Institute of Physics Publishing Ltd. (IOP); and Boca Raton, FL: CRC Press.

Balling, R. J., & Wilson, S. A. (2001). The maximin fitness function for multi-objective evolutionary computation: application to city planning. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*. Morgan Kaufmann Publishers Inc, San Francisco, CA, USA.

Bansal, J. C., Sharma, H., & Jadon, S. S. (2013). Artificial bee colony algorithm: a survey. *International Journal of Advanced Intelligence Paradigms, 5*(1-2),123-159.

Bellazzi, R., & Zupan, B. (2008). Predictive data mining in clinical medicine: current issues and guidelines. *International Journal of Medical Informatics, 77*(2), 81-97.

Burges, C. J. C. (1996). Simplified support vector decision rules. In *ICML'96 Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery, 2*(2), 121–167.

Burges, C. J. C., & Schölkopf, B. (1996). Improving the accuracy and speed of support vector machines. In *NIPS'96 Proceedings of the 9th International Conference on Neural Information Processing Systems*. MIT Press, Cambridge, MA, USA.

Charles, V., & Gherman, T. (2013). Achieving competitive advantage through big data. Strategic implications. *Middle-East Journal of Scientific Research, 16*(8), 1069-1074.

Charles, V., & Emrouznejad, A. (2018). Big Data for the Greater Good: An Introduction. In A. Emrouznejad & V. Charles (eds.), *Big Data for the Greater Good* (pp. 1-18). Switzerland: Springer Nature.

Charles, V., & Gherman, T. (2018). Big Data Analytics and Ethnography: Together for the Greater Good. In A. Emrouznejad & V. Charles (eds.), *Big Data for the Greater Good* (pp. 19-34). Switzerland: Springer Nature.

Charles, V., Tavana, M., & Gherman, T. (2015). The right to be forgotten – is privacy sold out in the big data age? *International Journal of Society Systems Science, 7*(4), 283-298.

Chen, J., Zhang, C., Xue, X., & Liu, C. L. (2013). Fast instance selection for speeding up support vector machines. *Knowledge-Based Systems, 45*, 1-7.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*(3), 273-297.

Czarnecki, W. M., & Tabor, J. (2014). Two ellipsoid Support Vector Machines. *Expert Systems with Applications, 41*, 8211-8224.

Das, S., Mullick, S. S., & Suganthan, P. N. (2016). Recent advances in differential evolution-An updated survey. *Swarm and Evolutionary Computation, 27*, 1-30.

De Jong, K. A. (2006). *Evolutionary Computation: A Unified Approach*. Cambridge, MA: The MIT Press.

De Jong, K., Fogel, D. B., & Schwefel, H. P. (1997). A history of evolutionary computation. In T. Bäck, D. B. Fogel, & Z. Michalewicz (eds.), *Handbook of evolutionary computation* (pp. A2.3:1-12). New York, NY: Oxford University Press; Bristol, UK: Institute of Physics Publishing Ltd. (IOP); and Boca Raton, FL: CRC Press.

Dezfoulian, M. H., MiriNezhad, Y., Mousavi, S. M. H., Mosleh, M. S., & Shalchi, M. M. (2016). Optimization of the Ho-Kashyap classification algorithm using appropriate learning samples. Paper presented at the *8th International Conference on Information and Knowledge Technology*. Hamedan, Iran.

Ding, Z., Huang, M., & Lu, Z. (2016). Structural damage detection using artificial bee colony algorithm with hybrid search strategy. *Swarm and Evolutionary Computation, 28*, 1-13.

Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms* (Ph.D. Thesis). Politecnico di Milano, Italy.

Downs, T., Gates, K. E., & Masters, A. (2001). Exact simplification of support vector solutions. *Journal of Machine Learning Research, 2*, 293-297.

Draa, A., & Bouaziz, A. (2014). An artificial bee colony algorithm for image contrast enhancement. *Swarm and Evolutionary Computation, 16*, 69-84.

Drugan, M. M. (2018). Reinforcement learning versus evolutionary computation: A survey on hybrid algorithms. *Swarm and Evolutionary Computation.* In press.

Duan, H.-b., Xu, C.-f., & Xing, Z.-H. (2010). A hybrid artificial bee colony optimization and quantum evolutionary algorithm for continuous optimization problems. *International Journal of Neural Systems, 20*(1), 39-50.

Dunn, J. C. (1973). A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics, 3*(3), 32-57.

Eiben, A. E., & Smith, J. E. (2003). *Introduction to Evolutionary Computing*. Berlin, Heidelberg: Springer-Verlag.

Emrouznejad, A., & Charles, V. (Eds.). (2018). *Big Data for the Greater Good*. Switzerland: Springer Nature.

Erol, O. K., & Eksin, I. (2006). A new optimization method: Big Bang-Big Crunch. *Advances in Engineering Software, 37*, 106-111.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Human Genetics, 7*(2), 179-188.

Fogel, D. B. (1997). Evolutionary algorithms in theory and practice. *Complexity, 2*(4), 26-27.

Fogel, D. B. (2000). Introduction to evolutionary computation. *Evolutionary Computation, 1*, 1-3.

Forina, M. *et al.* (1988). *PARVUS - An Extendible Package for Data Exploration, Classification and Correlation*. Institute of Pharmaceutical and Food Analysis and Technologies, Genoa, Italy.

Geem, Z. W., Kim, J. H., & Loganathan, G. V. (2001). A new heuristic optimization algorithm: harmony search. *Simulation, 76*(2), 60-68.

Gillespie, L. E., Gonzalez, G. R., & Schrum, J. (2017). Comparing direct and indirect encodings using both raw and hand-designed features in Tetris. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 179-186.

Global Pulse. (2012). *"Big Data for Development: Opportunities & Challenges": A Global Pulse White Paper*. Retrived from http://unglobalpulse.org/BigDataforDevWhitePaper.

Haberman, S. J. (1976). Generalized residuals for log-linear models. In *Proceedings of the 9th International Biometrics Conference* (pp. 104-122). Boston, MA, USA.

Hanley, J. A., & McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology, 143*(1), 29-36.

Ho, Y-C., & Kashyap, R. L. (1965). An algorithm for linear inequalities and its applications. *IEEE Transactions on Electronic Computers, 14*(5), 683-688.

Hsieh, T.-J., Hsiao, H.-F., & Yeh, W.-C. (2011). Forecasting stock markets using wavelet transforms and recurrent neural networks: an integrated system based on artificial bee colony algorithm. *Applied Soft Computing, 11*(2), 2510-2525.

Hsu, C. W., Chang, C. C., Lin, C. J., et al. (2003). *A practical guide to support vector classification.* Retrieved from http://www.csie.ntu.edu.tw/cjlin/libsvm

Justesen, N., & Risi, S. (2017). Continual online evolutionary planning for in-game build order adaptation in starcraft. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 187-194.

Kahraman, H. T., Sagiroglu, S., & Colak, I. (2013). The development of intuitive knowledge classifier and the modeling of domain dependent data. *Knowledge-Based Systems, 37*, 283-295.

Kallel, L., Naudts, B., & Rogers, A. (2001). *Theoretical Aspects of Evolutionary Computing*. Berlin, Heidelberg: Springer-Verlag.

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. *Technical Report-TR06*. Erciyes University, Turkey.

Karaboga, D., & Akay, B. (2011). A modified artificial bee colony (abc) algorithm for constrained optimization problems. *Applied Soft Computing, 11*(3), 3021-3031.

Karaboga, D., & Ozturk, C. (2010). Fuzzy clustering with artificial bee colony algorithm. *Scientific Research and Essays, 5*(14), 1899-1902.

Karaboga, D., Gorkemli, B., Ozturk, C., & Karaboga, N. (2014). A comprehensive survey: artificial bee colony (abc) algorithm and applications. *Artificial Intelligence Review, 42*(1), 21-57.

Kashan, M. H., Nahavandi, N., & Kashan, A. H. (2012). Disabc: a new artificial bee colony algorithm for binary optimization. *Applied Soft Computing, 12*(1), 342-352.

Keerthi, S. S., Chapelle, O., & DeCoste, D. (2006). Building support vector machines with reduced classifier complexity. *Journal of Machine Learning Research, 7*, 1493-1515.

Kennedy, J. (2017). Particle swarm optimization. In C. Sammut & G. I. Webb (eds.), *Encyclopedia of Machine Learning and Data Mning* (pp. 967-972). New York, NY: Springer.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In Proceedings of the IEEE International Conference Neural Networks, Australia, pp. 1942–1948.

Kiran, M. S. (2015). The continuous artificial bee colony algorithm for binary optimization. Applied Soft Computing. 33, 15-23.

Koh, H. C., & Tan, G. (2011). Data mining applications in healthcare. *Journal of Healthcare Information Management, 19*(2), 64-72.

Kumbhar, P. Y., & Krishnan, S. (2011). Use of artificial bee colony (ABC) algorithm in artificial neural network synthesis. International Journal of Advanced Engineering and Technology, 11(1), 162-171.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*, 436-444.

Lee, W., & Stolfo, S. J. (1998). Data Mining Approaches for Intrusion Detection. In *Proceedings of the 7th USENIX Security Symposium*. San Antonio, Texas.

Lee, J., Wu, F., Zhao, W., Ghaffari, M., Liao, L., Siegel, D. (2014). Prognostics and health management design for rotary machinery systems—Reviews, methodology and applications. *Mechanical Systems and Signal Processing, 42*(1-2), 314-334.

Liang, Z., Hu, K., Zhu, Q., & Zhu, Z. (2017). An enhanced artificial bee colony algorithm with adaptive differential operators. *Applied Soft Computing, 58*, 480-494.

Lloyd, S. P. (1982). Least square quantization in PCM. *IEEE Transactions of Information Theory, 28*(2), 129-137.

Luo, J., Wu, M., Gopukumar, D., & Zhao, Y. (2016). Big Data Application in Biomedical Research and Health Care: A Literature Review. *Biomedical Informatics Insights, 8*, 1-10.

Mavrovouniotis, M., Li, C., & Yang, S. (2017). A survey of swarm intelligence for dynamic optimization: Algorithms and applications. *Swarm and Evolutionary Computation, 33*, 1-17.

Metlicka, M., & Davendra, D. (2015). Chaos driven discrete artificial bee algorithm for location and assignment optimisation problems. *Swarm and Evolutionary Computation, 25*, 15-28.

Mirinezhad, S. Y., Dezfoulian, M. H., Mosleh, M. S., & Mousavi, S. H. (2017). Runtime Optimization of Widrow-Haff Classification Algorithm Using Proper Learning Samples. Paper presented at the *4th National Conference on Information Technology, Computer & Telecommunication*. Mashhad, Iran.

Mitchell, M. (1998). *An introduction to genetic algorithms*. Cambridge, MA: The MIT Press.

Morris, H. D., Feblowitz, J., Torchia, M., Ellis, S., & Knickle, K. (2014). A software platform for operational technology innovation. *International Data Corporation*, 1-17. Retrieved from

https://www.ge.com/digital/sites/default/files/download_assets/IDC_OT_Final_whitepaper_249120.pdf

Mousavi, S. M. H., MiriNezhad, S. Y., & Dezfoulian, M. H. (2017). Galaxy Gravity Optimization (GGO): An Algorithm for Optimization, Inspired by Comets Life Cycle. *Artifical Intelligence and Signal Processing (AISP)*, 1-10.

Mousavi, S. M. H., MiriNezhad, S. Y., & Mirmoini, A. (2017). A new support vector finder method, based on triangular calculations and K-means clustering. Paper presented at the *9th International Conference on Information and Knowledge Technology (IKT 2017)*. Tehran, Iran.

Nguyen, D., & Ho, T. (2006). A bottom-up method for simplifying support vector solutions. *IEEE Transactions on Neural Networks, 17*(3), 792-796.

Parhi, K. K. (2007). *VLSI Digital Signal Processing Systems: Design and Implementation*. New York, NY: John Wiley & Sons.

Parker, A., & Nitschke, G. S. (2017). Autonomous intersection driving with neuro-evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, Evolutionary Machine Learning Workshop, pp. 133-134.

Parzen, E. (1962). On estimation of a probability density function and mode. *The Annals of Mathematical Statistics, 33*(3), 1065-1076.

Piatetsky-Shapiro, G. (1999). The data-mining industry coming of age. *IEEE Intelligent Systems and Their Applications, 14*(6), 32-34.

Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, & A. J. Smola (eds.), *Advances in Kernel Methods: Support Vector Learning* (pp. 185-208). Cambridge, MA: MIT Press Cambridge.

Qi, K., Yang, H., Hu, Q., & Yang, D. (2019). A new adaptive weighted imbalanced data classifier via improved support vector machines with high-dimension nature. *Knowledge-Based Systems, 185*, 1-12.

Reed, P. M., Hadka, D., Herman, J. D., Kasprzyk, J. R., & Kollat, J. B. (2013). Evolutionary multiobjective optimization in water resources: the past, present, and future. *Advances in Water Resources, 51*, 438-456.

Rekha, A. G., Abdulla, M. S., & Asharaf, S. (2017). Lightly trained support vector data description for novelty detection. *Expert Systems with Applications, 85*, 25-32.

Rösler, O., & Suendermann, D. (2013). A First Step Towards Eye State Prediction Using EEG. In *Proceedings of the AIHLS 2013, International Conference on Applied Informatics for Health and Life Siences*. Istanbul, Turkey.

Saad, A., Khan, S. A., & Mahmood, A. (2018). A multi-objective evolutionary artificial bee colony algorithm for optimizing network topology design. *Swarm and Evolutionary Computation, 38*, 187-201.

Schölkopf, B., Smola, A. J., Williamson, R. C., & Bartlett, P. L. (2000). New support vector algorithms. *Neural Computation, 12*(5), 1207-1245.

Sigillito, V. G., Wing, S. P., Hutton, L. V. & Baker, K. B. (1989). Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest, 10*(3), 262-266.

Spirov, A., & Holloway, D. (2013). Using evolutionary computations to understand the design and evolution of gene and cell regulatory networks. *Methods, 62*(1), 39-55.

Steinbuch, K., & Widrow, B. (1965). A Critical Comparison of Two Kinds of Adaptive Classification Networks. *IEEE Transactions on Electronic Computers, 14*(5), 737-740. Originally appeared as a Stanford Electronics Laboratories Technical Report 6764-2, December 1964. Available at http://www-isl.stanford.edu/~widrow/papers/j1965acritical.pdf

Storn, R., & Price, K. (1997). Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*(4), 341-359.

Syed, N. A., Huan, S., Kah, L., & Sung, K. (1999). Incremental learning with support vector machines. In *Proceedings of the Workshop on Support Vector Machines at the International Joint Conference on Articial Intelligence* (IJCAI-99). Stockholm, Sweden.

Takagi, H. (2001). Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. *Proceedings of the IEEE, 89*(9), 1275-1296.

Theodoridis, S., & Koutroumbas, K. (2003). *Pattern recognition* (2nd ed). Boston, MA: Academic Press.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory* (2nd ed.). New York, NY: Springer-Verlag.

Yagain, D., & Vijayakrishna, A. (2015). A novel framework for retiming using evolutionary computation for high level synthesis of digital filters. *Swarm and Evolutionary Computation, 20*, 37-47.

Yang, X.-S. (2010). A new metaheuristic bat-inspired algorithm. In J. R. Gonzalez, D. A. Pelta, C. Cruz, G. Terrazas, & N. Krasnogor (eds.), *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)* (pp. 65-74). Berlin, Heidelberg: Springer.

Yang, X.-S., & Deb, S. (2009). Cuckoo search via Lévy flights. In *Proceedings of World Congress on Nature & Biologically Inspired Computing*. IEEE Publications, USA.

Yeh, I.-C., Yang, K.-J., & Ting, T.-M. (2009). Knowledge discovery on RFM model using Bernoulli sequence. *Expert Systems with Applications, 36*(3), 5866-5871.

Yu, Z., Jinhai, L., Guochang, G., Rubo, Z., Haiyan, Y. (2002). An implementation of evolutionary computation for path planning of cooperative mobile robots. In *Proceedings of the 4th World Congress on Intelligent Control and Automation*. IEEE, Shanghai, China.

Zhu, X-Y., Basten, T., Geilen, M., & Stuijk, S. (2012). Efficient retiming of multirate dsp algorithms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 31*(6), 831-844.

---------- Forwarded message ---------
From: **Binshan Lin** <eesserver@eesmail.elsevier.com>
Date: Wed, Feb 5, 2020 at 2:40 PM
Subject: Your Submission ESWA-D-18-04854R1

Ms. Ref. No.: ESWA-D-18-04854R1
Title: An Evolutionary Pentagon Support Vector Finder Method
Expert Systems With Applications or its open access mirror

Dear Dr. Vincent Charles,

Congratulations. I'm pleased to inform you that I have accepted your paper for publication in ESWA or ESWA X. Your accepted manuscript will now be transferred to our production department and work will begin on creation of the proof. If we need any additional information to create the proof, we will let you

know. If not, you will be contacted again in the next few days with a request to approve the proof and to complete a number of online forms that are required for publication. Your paper should appear as an "article in proofs" within two weeks of acceptance on ScienceDirect, and your printed version should appear in the journal within 2-3 months. Please note that the authors' affiliations must be the institutions where the research presented in the article took place; therefore, changes to the author affiliations are not permitted once the corrected proof is published online. Congratulations on your paper acceptance.

* ESWA has an Impact Factor of 4.292, based on the Journal Citation Reports 2018 by by Clarivate Analytics (released in June 2019).

* ESWA has a CiteScore 2018 of 6.36 (released in May 2019).

Look forward to your continued contributions to the ESWA/ESWA X in the near future.

With kind regards,

Dr. Binshan Lin
BellSouth Professor
Editor-in-Chief, Expert Systems with Applications